Freescale Semiconductor
Application Note

Document Number: AN4647
Rev. 0, 1/2013

Configuring and Using the 2D-ACE on Vybrid Microcontrollers

by: Steve McAslan

Contents

1 Introduction....1
2 Hardware interface....1
3 Enabling the DCUv4....2
3.1 Choosing a suitable clock source. 2
3.2 Making the TCON module active on the DCUv4....3
4 Configuring the DCUv4....3
4.1 Configuring the DCUv4 for a TFT LCD panel without TCON....3
4.2 Configuring the timing parameters....5
4.3 Transferring the configuration. 7
4.4 Enabling and configuring TCON settings....8
4.5 Testing the hardware and software configuration....8
5 Using the DCUv4....9
5.1 Using the graphic layers....10
5.2 Error reporting....12
5.3 Using the cursor....13
6 Summary....14

1 Introduction

The Vybrid family of microcontrollers features up to two independent display peripherals that allow it to drive up to two TFT LCD panels. The 2D-ACE module features a rich set of capabilities that allow you to build engaging and dynamic graphic content from pre-rendered images and display this content on a wide range of TFT LCD panels. This application note explains the steps to configure and use the 2D-ACE with TFT LCD panels and how to display and blend graphics on each panel. See freescale.com for more details on how to use these features in a typical application.

The hardware module that implements the 2D-ACE functionality in the Vybrid family is known as the Display Control Unit version 4 (DCUv4) and this how the MCU documentation and software examples address the module. In this application note, the term DCUv4 is used to refer the general features of the module, while individual implementations are referred to as DCU0 or DCU1.

2 Hardware interface

In most cases, the DCUv4 connects directly to the TFT LCD panel through the MCU I/O pins. The DCUv4 can connect to a panel that includes a timing controller (TCON) and uses a

parallel RGB interface. Alternatively, it can connect to a panel that includes a TCON. In this case, the pin connections are different.

It is possible to use panels that accept less than 24-bit data input; in this case, it is normal to connect the most significant bits from the DCUv4 to the panel data pins. For example, in the case of an 18-bit panel (RGB each with 6-bits) it is normal to connect as shown in Table 1.

NOTE

The unused pins may be used as GPIOs or other functions where available.

Table 1. Recommended connection for 18-bit panels

MCU = panel	MCU = panel	MCU = panel
R7 = R5	G7 = G5	B7 = B5
R6 = R4	G6 = G4	B6 = B4
R5 = R3	G5 = G3	B5 = B3
R4 = R2	G4 = G2	B4 = B2
R3 = R1	G3 = G1	B3 = B1
R2 = R0	G2 = G0	B2 = B0
R1 unused	G1 unused	B1 unused
R0 unused	G0 unused	B0 unused

Other timing signals are connected to the panel as required. For example, some panels do not require a horizontal sync signal and, in this case, the pin can be used as GPIOs.

3 Enabling the DCUv4

To enable each DCUv4, simply supply it with an appropriate clock. It is possible to supply both DCU0 and DCU1 with the same clock or with different clocks.

3.1 Choosing a suitable clock source

The DCUv4 takes an input clock, divided from one of two source clocks, and further divides it inside the module to produce the pixel clock required by the connected TFT LCD panel. The choice of clocks and multiple dividers allows a very wide range of pixel clock values to be derived. This is an important feature because a typical TFT LCD panel needs to be refreshed at approximately 60 Hz independent of its size. The greater the number of pixels on the panel, the faster the pixel clock has to be to transfer each pixel in the available time; thus, a wide range of clock values must be supported.

Either DCUv4 can choose from the PLL1 PFD2 clock, which has a default value of 452 MHz, or the PLL3 (USB) clock, which is configured for 480 MHz. This clock is then divided by any integer value between 1 and 8.

Configure the clock source such that it can give the best match to the required pixel clock when divided by an integer value. For example, a QVGA panel requires a pixel clock of approximately 5.33 MHz. If the selected clock is PLL3 divided by 5 (96 MHz), then a divider of 18 gives a close match to this value. A wide QVGA panel (480 × 272) requires a pixel clock of approximately 9 MHz. If the selected clock is PLL1 PFD2 divided by 5 (90.4 MHz), then a divider of 10 gives a close match. Panels typically have a wide operating range and so finding a compatible clock value is usually straightforward.

Select the clocks for each DCUv4 by using the DCU0_CLK_SEL and DCU1_CLK_SEL bits in the CCM_CSCMR1 register. Predivide the source clock to the DCUv4s by writing the appropriate value to the DCU0_DIV and DCU1_DIV fields in the CCM_CSCDR3 register. Finally, enable the clocks to the modules by using the DCU0_EN and DCU1_EN bits in the CCM_CSCDR3 register. It is permissible to write to all the fields in the CCM_CSCDR3 register in a single operation if required.

3.2 Making the TCON module active on the DCUv4

When the external panel does not contain its own Timing Controller (TCON), use the Vybrid TCON module. If the TCON module is not required, then bypass it by setting the TCON_BYPASS bit in the TCON_CTRL1 register—at reset the module is not bypassed. See application note AN4635, *Using the Vybrid TCON Module*, for a detailed discussion of how to use the TCON.

4 Configuring the DCUv4

When the DCUv4 has a suitable clock, it is possible to begin configuring the module for the TFT LCD panel being used and to verify that the connections are operating as expected. Note that the TCON module is connected by default and therefore if it is not required it must be bypassed—see Making the TCON module active on the DCUv4 for more details.

4.1 Configuring the DCUv4 for a TFT LCD panel without TCON

First, configure the I/O pads so that the DCUv4 function is selected. This is done by modifying the IOMUX Pad Control register associated with each pin.

The most relevant bit field in the Pad Control Register is the MUX mode select (MUX_MODE) field. As soon as the DCUv4 function is selected, the pin becomes an output controlled by the DCUv4. Other bit fields control the behavior of the pad with respect to the physical requirements of the board and the speed of the DCU signal. Configure the SPEED, SRE, and DSE bit fields to optimize signal performance with maximum EMC characteristics. The values chosen depend entirely on the size of the panel and characteristics of the application printed-circuit board.

There are a wide range of multiplexing options on the Vybrid microntroller, so it is impossible to give specific configuration settings for every option. The following tables give the configuration for the most likely pins used for each DCUv4 module. Adjust these settings to suit your own hardware design.

To enable the DCU0 signals on the primary multiplex setting, configure the Vybrid IOMUX Pad Control registers as given in Table 2. The configuration values for input and pull resistors are disabled, but they can be enabled if required. The slew is set to slow, the DSE setting is 50 ohms, and the SPEED setting is medium; these must be configured to optimize the performance of the board and the panel.

Table 2. IOMUX Pad Control Register example values for DCU0

Signal	Register	MUX_ MODE	SPEED	SRE	ODE	HYS	DSE	PUS	PKE	PUE	OBE	IBE
R0	110	1	1	0	0	0	3	0	0	0	X	0
R1	111	1	1	0	0	0	3	0	0	0	X	0
R2	112	1	1	0	0	0	3	0	0	0	X	0
R3	113	1	1	0	0	0	3	0	0	0	X	0
R4	114	1	1	0	0	0	3	0	0	0	X	0

Table continues on the next page...

Table 2. IOMUX Pad Control Register example values for DCU0 (continued)

Signal	Register	MUX_ MODE	SPEED	SRE	ODE	HYS	DSE	PUS	PKE	PUE	OBE	IBE
R5	115	1	1	0	0	0	3	0	0	0	X	0
R6	116	1	1	0	0	0	3	0	0	0	X	0
R7	117	1	1	0	0	0	3	0	0	0	X	0
G0	118	1	1	0	0	0	3	0	0	0	X	0
G1	119	1	1	0	0	0	3	0	0	0	X	0
G2	120	1	1	0	0	0	3	0	0	0	X	0
G3	121	1	1	0	0	0	3	0	0	0	X	0
G4	122	1	1	0	0	0	3	0	0	0	X	0
G5	123	1	1	0	0	0	3	0	0	0	X	0
G6	124	1	1	0	0	0	3	0	0	0	X	0
G7	125	1	1	0	0	0	3	0	0	0	X	0
B0	126	1	1	0	0	0	3	0	0	0	X	0
B1	127	1	1	0	0	0	3	0	0	0	X	0
B2	128	1	1	0	0	0	3	0	0	0	X	0
B3	129	1	1	0	0	0	3	0	0	0	X	0
B4	130	1	1	0	0	0	3	0	0	0	X	0
B5	131	1	1	0	0	0	3	0	0	0	X	0
B6	132	1	1	0	0	0	3	0	0	0	X	0
B7	133	1	1	0	0	0	3	0	0	0	X	0
VSYNC	106	1	1	0	0	0	3	0	0	0	X	0
HSYNC	105	1	1	0	0	0	3	0	0	0	X	0
DE	109	1	1	0	0	0	3	0	0	0	X	0
PCLK	107	1	1	1	0	0	3	0	0	0	X	0

To enable the DCU1 signals on a valid multiplex setting, configure the Vybrid IOMUX Pad Control registers as given in Table 3. The configuration values for input and pull resistors are disabled, but they can be enabled if required. The slew is set to slow, the DSE setting is 50 ohms, and the SPEED setting is medium; these must be configured to optimize the performance of the board and the panel.

Table 3. IOMUX Pad Control Register example values for DCU1

Signal	Register	MUX_ MODE	SPEED	SRE	ODE	HYS	DSE	PUS	PKE	PUE	OBE	IBE
R0	74	7	1	0	0	0	3	0	0	0	X	0
R1	75	7	1	0	0	0	3	0	0	0	X	0
R2	0	7	1	0	0	0	3	0	0	0	X	0
R3	71	7	1	0	0	0	3	0	0	0	X	0
R4	72	7	1	0	0	0	3	0	0	0	X	0
R5	73	7	1	0	0	0	3	0	0	0	X	0
R6	30	7	1	0	0	0	3	0	0	0	X	0

Table continues on the next page...

Table 3. IOMUX Pad Control Register example values for DCU1 (continued)

Signal	Register	MUX_ MODE	SPEED	SRE	ODE	HYS	DSE	PUS	PKE	PUE	OBE	IBE
R7	31	7	1	0	0	0	3	0	0	0	X	0
G0	76	7	1	0	0	0	3	0	0	0	X	0
G1	77	1	1	0	0	0	3	0	0	0	X	0
G2	78	7	1	0	0	0	3	0	0	0	X	0
G3	93	7	1	0	0	0	3	0	0	0	X	0
G4	94	7	1	0	0	0	3	0	0	0	X	0
G5	95	7	1	0	0	0	3	0	0	0	X	0
G6	96	7	1	0	0	0	3	0	0	0	X	0
G7	97	7	1	0	0	0	3	0	0	0	X	0
B0	88	7	1	0	0	0	3	0	0	0	X	0
B1	89	7	1	0	0	0	3	0	0	0	X	0
B2	100	7	1	0	0	0	3	0	0	0	X	0
B3	101	7	1	0	0	0	3	0	0	0	X	0
B4	102	7	1	0	0	0	3	0	0	0	X	0
B5	103	7	1	0	0	0	3	0	0	0	X	0
B6	104	7	1	0	0	0	3	0	0	0	X	0
B7	99	7	1	0	0	0	3	0	0	0	X	0
VSYNC	11	7	1	0	0	0	3	0	0	0	X	0
HSYNC	10	7	1	0	0	0	3	0	0	0	X	0
DE	13	7	1	0	0	0	3	0	0	0	X	0
PCLK	43	7	1	1	0	0	3	0	0	0	X	0

4.2 Configuring the timing parameters

The second step is to configure the DCUv4 operating parameters to match the specification of the panel being used. The relevant configuration registers are given in Table 4.

Table 4. DCUv4 panel configuration registers

Register	Function
DIV_RATIO	Divides the selected DCUv4 auxiliary clock down to the required pixel clock value
DISP_SIZE	Defines the number of horizontal and vertical pixels on the panel
HSYN_PARA	Defines the horizontal (line) timing parameters
VSYN_PARA	Defines the vertical (whole frame) timing parameters
SYN_POL	Defines the polarity of the timing signals

Start by providing the correct division ratio for the auxiliary clock provided to the DCU. This is a simple modulus division with the value in DIV_RATIO being one less than the divider required; for example, a value of 0 gives a division of 1.

Table 5 shows some example DIV_RATIO values for different panel sizes. The table assumes the following pixel clock requirements for the example panels: WQVGA (480 × 272) = 9 MHz and WQVGA (800 × 480) = 33 MHz. Note that the chosen DIV_RATIO value will typically not be able to match the target pixel clock frequency exactly and so judgment must be used to verify that the value is suitable for the panel.

Note that in all cases the DCU should operate at a factor above the pixel clock since it may have to fetch multiple graphics for each on-panel pixel (for example, it is possible to have up to six 32-bit per pixel source images that overlap to produce a single 24-bit output pixel). A ratio of 4 or more is a good starting point in most applications.

Table 5. Example DIV_RATIO values for different panels

Chosen clock (MHz)	WQVGA	WVGA
240	45	7
160	30	5
120	22	X
96	18	3

Next, configure the DISP_SIZE to match the panel size. The vertical dimension of the panel is entered directly in the DELTA_Y bit field. The horizontal dimension is configured differently. The DCUv4 requires the width of the panel to be a multiple of 16 and the DELTA_X bit field indicates the number of multiples in the horizontal dimension. In other words, the DELTA_X field contains the width of the panel divided by 16. See Table 6 for some typical examples.

Table 6. Example DISP_SIZE values for different panels

Panel size	DELTA_Y	DELTA_X
320H × 240V	240	20
240H × 320V	320	12
480H × 272V	272	24
640H × 240V	240	32

Now the synchronization signals must be configured for the panel. Typically, each panel has unique timing requirements and not all timing signals are required for all panels; however, manufacturers tend to specify the timing configuration in a consistent way that matches the parameter requirements for the DCUv4. In most cases, it is possible to read the timing parameter from a TFT LCD specification and enter that number directly into the relevant DCUv4 configuration register.

Begin the configuration with the HSYN_PARA register that configures the horizontal signals. Three values are required as illustrated in Figure 1 and all are defined in multiples of the pixel clock:

- Horizontal pulse width, which defines the number of pixel clocks for which the horizontal timing pulse is active (HSYNC signal).
- Horizontal back porch width, which defines the number of pixel clocks after the horizontal pulse before the start of pixel data. Some manufacturers define the back porch starting from the start of the horizontal pulse rather than the end.
- Horizontal front porch width, which defines the number of pixel clocks after the pixel data before the start of the horizontal pulse.

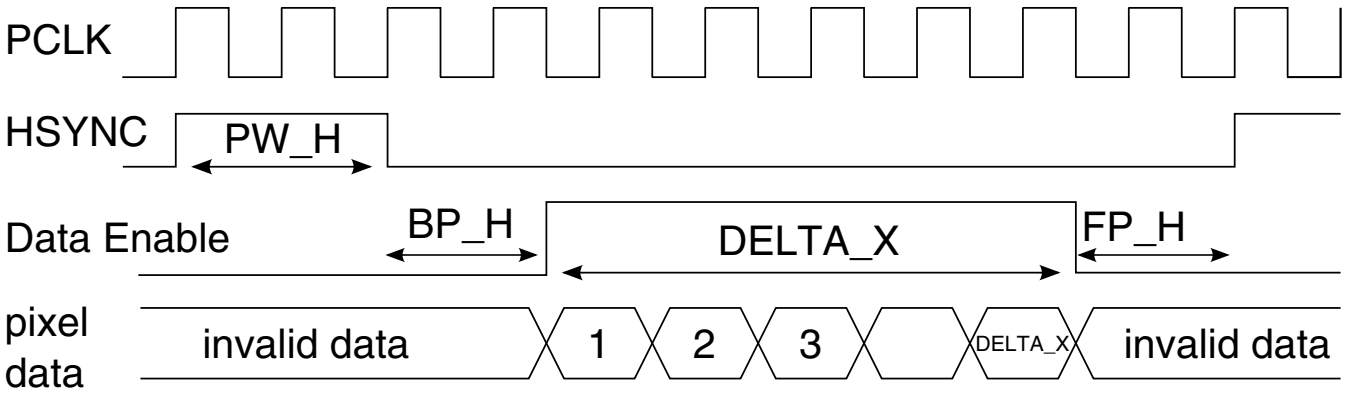

Figure 1. Horizontal timing diagram

Next, configure the vertical timing signals using the VSYN_PARA register. Three values are required as illustrated in Figure 2 and this time all are defined in multiples of horizontal lines:

- Vertical pulse width, which defines the number of horizontal lines for which the vertical timing pulse is active (VSYNC signal).
- Vertical back porch width, which defines the number of horizontal lines after the vertical pulse before the start of pixel data. Some manufacturers define the back porch starting from the start of the vertical pulse rather than the end.
- Vertical front porch width, which defines the number of horizontal lines after the pixel data before the start of the vertical pulse.

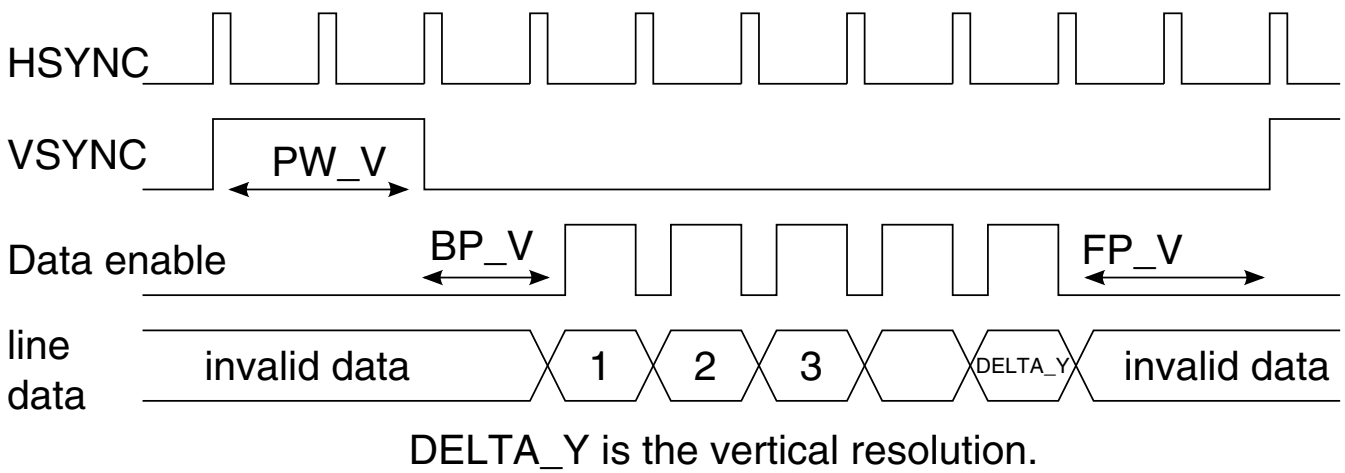

Figure 2. Vertical timing diagram

The polarity of the pixel clock (PCLK), the horizontal timing signal (HSYNC), and the vertical timing signal (VSYNC) can be inverted using the SYN_POL register.

4.3 Transferring the configuration

Before the DCUv4 will begin operating it must have the configuration written to its registers transferred to its internal state machine. After the first transfer, it is possible to have the transfer performed automatically each time the panel contents are updated (each vertical frame time). For simplicity it is recommended that the automatic update method is selected; this can be changed at any time, but remember that changes to the DCUv4 configuration—including layer configuration—must be transferred before they take effect.

When the DCUv4 configuration is complete, set the READREG bit in the UPDATE_MODE register. This will start the process of transferring the configuration. The process is complete when the READREG bit is cleared to 0. It is now possible to enable automatic updating by setting the MODE bit in the UPDATE_MODE register. When this bit is set, further changes to the configuration will be reflected in the next frame update to the panel.

4.4 Enabling and configuring TCON settings

If the TCON is in use, then there are further adjustments to be made. See AN4635, *Using the Vybrid TCON Module*, for details on how to configure the TCON.

4.5 Testing the hardware and software configuration

After the DCUv4 is enabled, configured, and the configuration is transferred, it is possible to display graphics on the panel or perform verification tests. Using the DCUv4 describes the normal use of the panel and this section provides guidance on how to verify the hardware and software configuration of the system.

The DCUv4 includes a special test-card mode that displays bands of known colors on the panel. This mode allows verification so that the connection to the panel and the DCUv4 timing configuration is correct.

At this stage, the panel is connected and the DCUv4 is configured to drive the panel but no timing signals are being driven by the Vybrid microcontroller. It is possible to start the pixel clock before HSYNC, VSYNC, and the pixel data; this is a requirement for some panels. It is also possible to start all signals at the same time.

To start the pixel clock before other timing signals, follow the given steps:

- In the DCU_MODE Register, set the DCU_MODE bit field to 3 which selects color bar mode.
- In the DCU_MODE Register, set the RASTER_EN bit field to 1 which enables the raster signals (HSYNC, VSYNC, pixel data).

To start all the signals together, follow this additional step:

- In the DCU_MODE Register, set the DCU_MODE bit-field to 3 and the RASTER_EN bit field to 1.

The DCUv4 will begin sending pixel data which divides the panel into 8 color bars horizontally. By default, each color bar is assigned a color as shown in Figure 3.

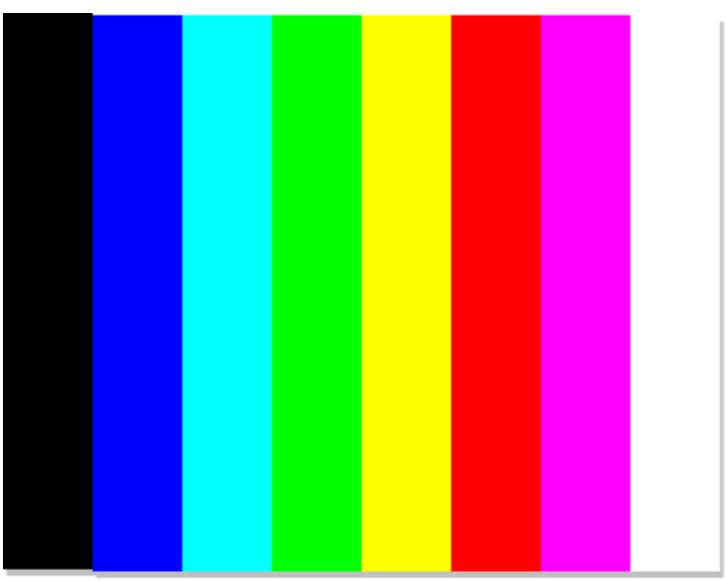

Figure 3. Default color bar mode configuration

This standard configuration can be used to verify that the connection to the panel and the timing are correct. Verify the timing by measuring the output signals and comparing them to the values expected by the panel. The pixel data connection integrity can be verified by examining the content of the panel. This setting also allows testing and configuration of the panel backlight control.

It is possible to verify the connection of each of the pixel data lines by modifying the color bar color settings. This is done using the COLBAR_1 to COLBAR_8 Registers. The registers are numbered to represent the bars from left to right so that COLBAR_1 contains the color setting of the left-hand bar (in the default case this is black) and COLBAR contains the color settings of the right-hand bar (white). Each COLBAR_n Register contains an RGB888 value with 8 bits to define the red, green, and blue content of the bar.It is possible to change these registers to any other value. As an example, it is possible to verify that each green data line is connected and that there are no short circuits by changing the COLBAR_n Registers to contain only a single green data line value as follows:

- Set COLBAR_1 to 0x00008000
- Set COLBAR_2 to 0x00004000
- Set COLBAR_3 to 0x00002000
- Set COLBAR_4 to 0x00001000
- Set COLBAR_5 to 0x00000800
- Set COLBAR_6 to 0x00000400
- Set COLBAR_7 to 0x00000200
- Set COLBAR_8 to 0x00000100

The resulting panel content is shown in Figure 4. Each color bar should be a different shade of green with no red or blue visible. This is not a comprehensive test on its own but combined with other combinations and test approaches may be used to visually and electrically verify correct configuration.

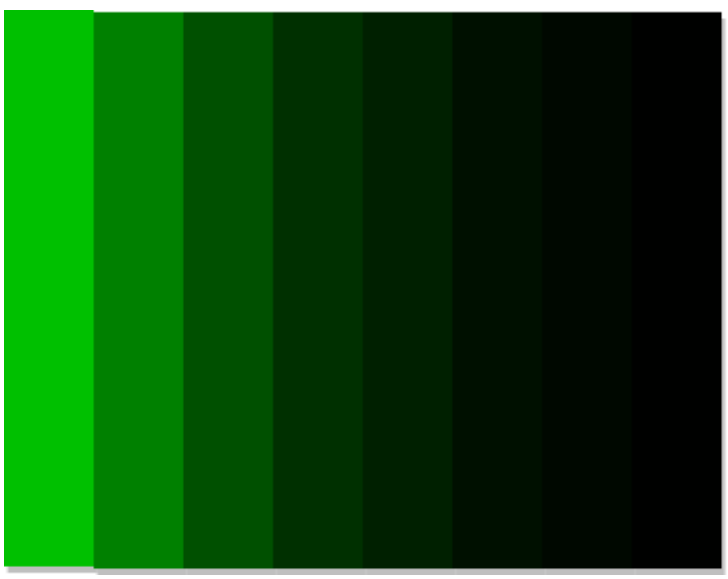

Figure 4. Green data line connection test (simulated)

5 Using the DCUv4

When the panel hardware and software configuration have been verified, it is possible to begin using the DCUv4 in its functional mode.

To enable the panel for normal use:

- In the DCU_MODE Register, set the DCU_MODE bit field to 1 which selects normal mode.

- In the DCU_MODE Register, set the RASTER_EN bit field to 1 which enables the raster signals (HSYNC, VSYNC, pixel data).
- OR to start all signals together, in the DCU_MODE Register set the DCU_MODE bit field to 1 and the RASTER_EN bit-field to 1.

At this point, the panel will be active and it will display a single color. This color is configured in the BGND register and is black after reset. Like the COLBAR_n Registers, this register can be modified to display any single color by modifying its value to another RGB888 value.

Functional content is placed on the panel by enabling graphic layers and the cursor. These elements are discussed in the following sections with a focus on their basic properties. To see examples of how an application may use the different features, refer to demonstration software, application notes, and video examples available at freescale.com.

5.1 Using the graphic layers

The most common way of presenting content on the panel is the layer. By configuring a layer it is possible to fetch a graphic directly from any memory-mapped module and display it on the panel. The graphic data may be encoded in many different formats. There are 64 layers in total and each is configured using 9 registers. The layers are independent of each other and placed in a fixed priority on the panel. The steps to display a graphic on a layer are as follows:

1. Configure the size of the layer using CtrlDescLn_1. This contains two 11-bit fields that describe the width and height of the layer that will appear on the panel. The width of the graphic must conform to configuration rules related to the graphic encoding as described in the Vybrid Reference Manual.

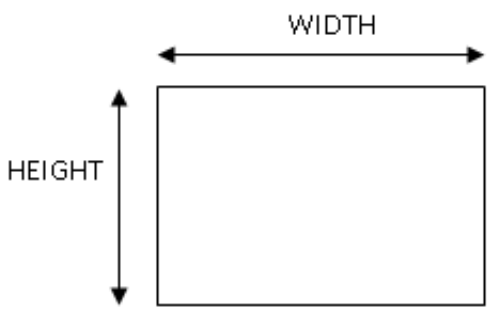

Figure 5. CTRLDESCLn_1 Register

2. Place the layer on the panel using CtrlDescLn_2. This contains two fields: POSX and POSY which indicate the (x,y) coordinate of the top left pixel. The values are both 12-bit signed integers. The POSX coordinate operates as a normal Cartesian coordinate in that negative values are to the left of the lefthand edge of the panel. The POSY coordinate operates as an inverted Cartesian coordinate; a negative value is above the top of the panel. Any coordinate location may be used, including values that are fully or partially beyond the extent of the screen. In this case, any portion of a layer not within the area of the panel is not displayed.

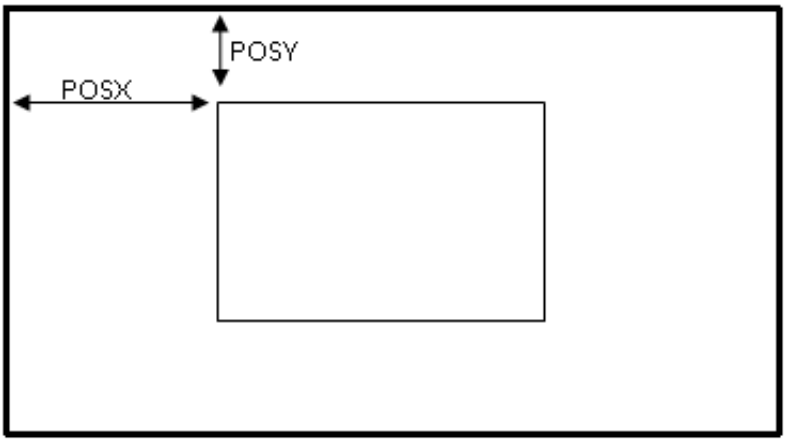

Figure 6. CTRLDESCLn_2 Register

3. Point to the top left-hand pixel of the graphic in memory using CtrlDescLn_3. This contains the 32-bit address of the graphic.

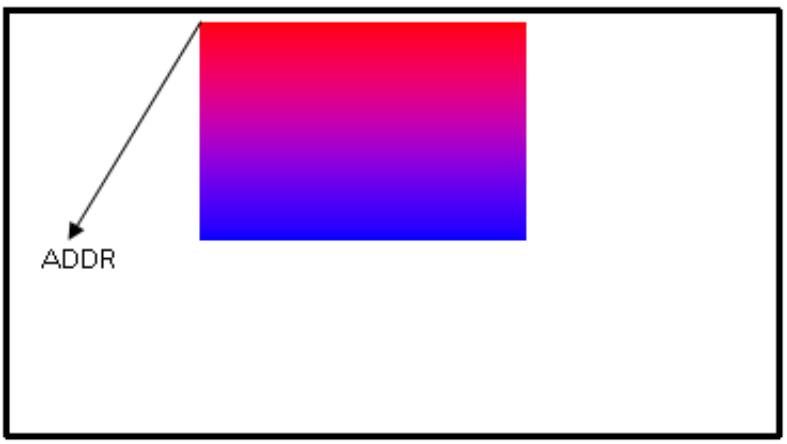

Figure 7. CTRLDESCLn_3 Register

4. Choose the graphic encoding, blending, and tiling and then enable it using CtrlDescLn_4. This register configures the appearance of the graphic on the panel and how it relates to other graphics.

Figure 8. CTRLDESCLn_4 Register

5. Configure the chroma-keying ranges using CtrlDescLn_5 and CtrlDescLn_6. These registers are used to define which colors of pixels are selected within the graphic. They define the maximum and minimum POSY POSX ADDR values for each color component of the selected pixels. The behavior of the selected pixels is controlled using CtrlDescLn_4. Figure 9 illustrates an example in which pixels of a certain color range are selected and then made partially transparent.

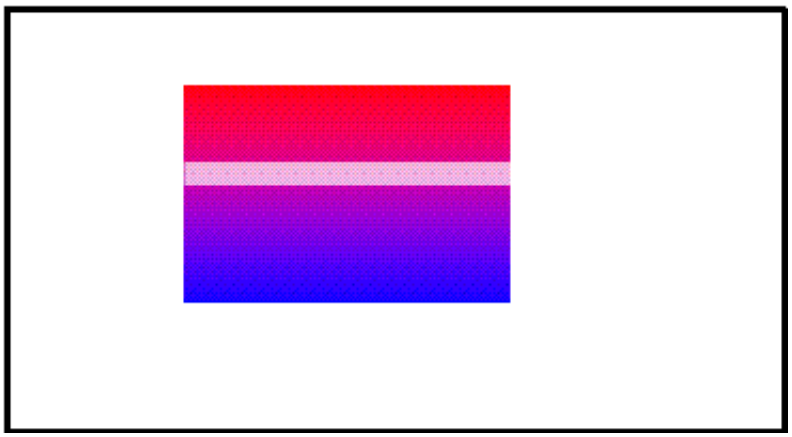

Figure 9. CTRLDESCLn_5 and CTRLDESCLn_6 Register

6. Configure the tile size using CtrlDescLn_7. If the graphic is to be used as a tile, then the size of the graphic on the panel is defined by the values in CtrlDescLn_1 and the size of the source graphic is defined by using the TILE_VER_SIZE (height) and TILE_HOR_SIZE (width) fields.

Figure 10. CTRLDESCLn_7 Register

7. Configure the transparency colors of the layer using CtrlDescLn_8 and CtrlDescLn_9. The DCUv4 supports two different transparency formats in which the graphic contains no color information. In this case, the color is provided by these two registers and the DCUv4 creates the final image by pre-blending the colors in these registers using the alpha information stored with the graphic. The final appearance again depends on the blending setting configured in the CtrlDescLn_4 register.

5.2 Error reporting

Errors in the configuration of layers are reported in the PARR_ERR_STATUS Registers. Note that an error in the layer configuration registers will prevent the layer from being visible. If a layer is not visible as expected, then check the flags in the PARR_ERR_STATUS Registers to see if an error has been detected.

The priority of the layers is fixed so that layer 0 is always higher priority than layer 1, which is higher than layer 2, and so on. This arrangement influences the effect of blending and means that some layers may be completely obscured by a higher priority layer.

5.3 Using the cursor

The cursor is a graphic element independent from the graphic layers. It allows a single graphic to be placed on top of all the layers and does not support any blending or choice of graphic encoding. The cursor is always stored in a special area of RAM with the DCUv4 module.

The following steps allow you to display the cursor:

1. Configure the size of the cursor using CtrlDescCursor_1. The maximum size of the cursor is determined by the size of the internal RAM.

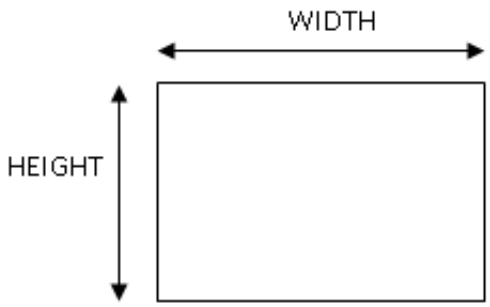

Figure 11. CtrlDescCursor_1 Register

2. Place the layer on the panel using CtrlDescCursor_2.

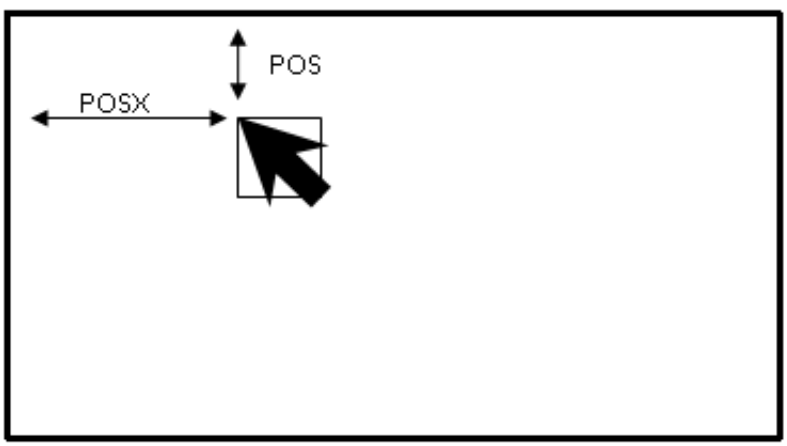

Figure 12. CtrlDescCursor_2 Register

3. Choose the color of the cursor using CtrlDescCursor_3.

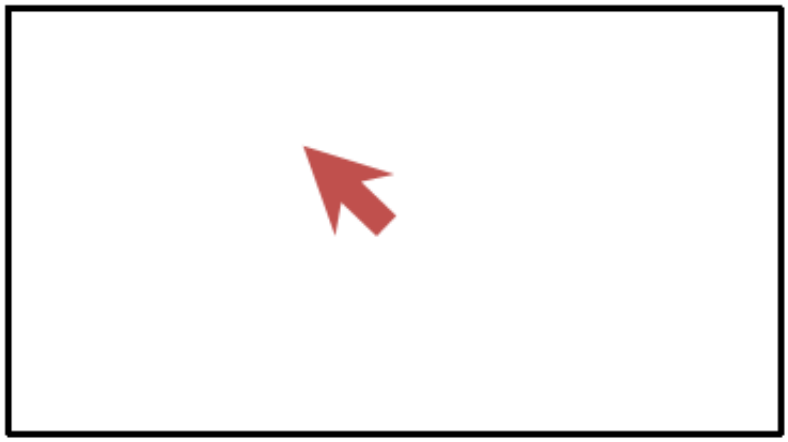

Figure 13. CtrlDescCursor_3 Register

4. Choose the blinking options using CtrlDescCursor_4.

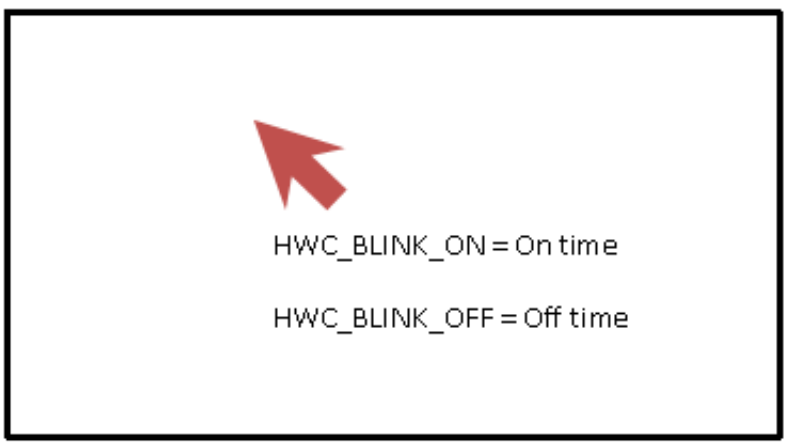

Figure 14. CtrlDescCursor_4 Register

6 Summary

This application note describes the steps for enabling and using the DCUv4 on the Vybrid family of microcontrollers. The nature of graphical user interfaces is such that layers of graphic software and individual image designs will be required to make use of the DCUv4 in a typical application. Examples of this software and design flow recommendations are available from the Freescale resource library available at freescale.com.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

Document Number: AN4647
Rev. 0, 1/2013

freescale